



## Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : <http://oatao.univ-toulouse.fr/>  
Eprints ID : 12621

**To cite this version** : Negre, Elsa and Ravat, Franck and Teste, Olivier and Tournier, Ronan *[Problème du démarrage à froid pour les systèmes de recommandation dans les entrepôts de données](#)*. (2013) In: INFormatique des Organisations et Systemes d'Information et de Decision - INFORSID 2013, 29 May 2013 - 31 May 2013 (Paris, France).

Any correspondance concerning this service should be sent to the repository administrator: [staff-oatao@listes-diff.inp-toulouse.fr](mailto:staff-oatao@listes-diff.inp-toulouse.fr)

---

# Problème du démarrage à froid pour les systèmes de recommandation dans les entrepôts de données

Elsa Negre<sup>\*</sup> — Franck Ravat<sup>\*\*</sup> — Olivier Teste<sup>\*\*\*</sup> — Ronan Tournier<sup>\*\*</sup>

<sup>\*</sup> Université Paris Dauphine, LAMSADE, Place du Maréchal de Lattre de Tassigny, 75116 Paris : Elsa.Negre@dauphine.fr

<sup>\*\*</sup> Université Toulouse 1 Capitole, IRIT-SIG, 2 rue du doyen Gabriel Marty, 31042 Toulouse Cedex 9 : Franck.Ravat@irit.fr ; Ronan.Tournier@irit.fr

<sup>\*\*\*</sup> Université Toulouse 2 - IUT Blagnac, IRIT-SIG, 1 place Georges Brassens, BP 60073, 31703 Blagnac Cedex : Olivier.Teste@irit.fr

---

**RÉSUMÉ.** L'exploration OLAP (On-Line Analytical Processing) de données est un processus incrémental basé sur des requêtes effectuant une recherche d'informations dans un Entrepôt de Données (ED) multidimensionnelles. Afin de faciliter l'exploration d'un décideur, des systèmes de recommandation ont été proposés. Toutefois, lors de l'utilisation d'un nouveau système, ces systèmes de recommandations ne fonctionnent plus (problème de démarrage à froid). Dans cet article, nous proposons des recommandations pour un décideur qui est confronté à un problème de démarrage à froid. Notre processus est composé de quatre étapes : identifier le squelette des requêtes OLAP, prévoir des opérations candidates, calculer des recommandations sur les opérations candidates et classer ces recommandations.

**ABSTRACT.** Exploring data is an incremental OLAP (On-Line Analytical Processing) query process for searching relevant information in a multidimensional Data Warehouse. In order to ease user exploration, recommender systems are used. However when facing a new system, such recommendations do not operate anymore. This is known as the cold-start problem. In this paper, we provide recommendations to the user while facing this cold-start problem in a new system. Our process is composed of four steps: patternizing queries, predicting candidate operations, computing candidate recommendations and ranking these recommendations.

**MOTS-CLÉS :** Démarrage à froid, système de recommandation, OLAP, entrepôt de données

**KEYWORDS:** Cold-start problem, Recommender system, OLAP, datawarehouse

---

## 1. Introduction et contexte

Les systèmes d'aide à la décision permettent aux décideurs d'analyser l'activité d'une entreprise ou d'une organisation en explorant des données historisées et agrégées d'un Entrepôt de Données (ED) [Colliat1996, Kimball1996]. Les décideurs explorent et analysent ces données en utilisant les processus On-Line Analytical Processing (OLAP). Cependant, l'exploration d'un ED volumineux peut s'avérer fastidieuse pour un décideur [Lyman *et al.*2003]. Ainsi, des techniques de recommandation facilitant l'exploration d'informations pertinentes sont nécessaires. Le processus de recommandation vise à guider l'utilisateur au cours de son exploration d'informations volumineuses vers les informations qui apparaissent comme les plus pertinentes.

Les systèmes de recommandation sont fréquents dans le domaine du commerce électronique [Adomavicius et Tuzhilin2005, Baeza-Yates2010]. Notamment, ils proposent un objet à un client en fonction de ses préférences. Toutefois, dans le domaine des bases de données, certains travaux [Chatzopoulou *et al.*2009, Khoussainova *et al.*2009, Stefanidis *et al.*2009] ont proposé des méthodes et des algorithmes pour aider l'utilisateur lors de l'élaboration de ses requêtes. Dans le cadre de la recommandation dans les ED à partir de requêtes OLAP (voir [Marcel et Negre2011] et [Negre2009] pour plus de détails), certains exploitent les profils et les préférences des décideurs [Jerbi *et al.*2009, Bellatreche *et al.*2005, Golfarelli *et al.*2011], d'autres utilisent la découverte de connaissances faites au cours des analyses décisionnelles [Sarawagi2000, Cariou *et al.*2008] ainsi que sur l'exploitation des journaux contenant des séquences de requêtes précédemment exécutés par d'autres utilisateurs sur le même cube de données [Sapia1999, Chatzopoulou *et al.*2009, Yang *et al.*2009, Giacometti *et al.*2009, Giacometti *et al.*2011]. Un cube de données est une représentation multidimensionnelle des données sur laquelle le décideur effectue ses analyses OLAP. Plus récemment, [Romero *et al.*2011] propose une algèbre multidimensionnelle pour décrire le processus analytique des sessions.

Bien que ces approches permettent de recommander des requêtes pertinentes à un utilisateur, elles s'avèrent inopérantes lorsque l'ED est mis à jour ou lorsque les décideurs sont différents ou nouveaux. Ce problème du démarrage à froid est rencontré lorsque les recommandations sont nécessaires pour des objets et / ou des utilisateurs pour lesquels nous n'avons aucune information explicite ou implicite [Schein *et al.*2001]. Il y a plusieurs problèmes liés au démarrage à froid [Kłopotek2008] : nouvel utilisateur [Golbandi *et al.*2011, Rashid *et al.*2008], nouveau produit [Schein *et al.*2001], association non transitive et apprentissage. Dans le contexte des ED, nous sommes confrontés au problème du démarrage à froid lorsque le système souhaite formuler des recommandations pour un nouveau cube de données. Par exemple, prenons le cas d'un ED contenant (1) des données jusqu'à l'année 2010, (2) un cube de données C1 portant sur les années 2009 et 2010 et (3) le journal des requêtes permettant de construire C1. Supposons que cet ED a été mis à jour pour intégrer les données de 2011 et qu'un nouveau cube C2 portant sur les années 2010

et 2011 a été créé, le système ne pourra faire que très peu de recommandations aux décideurs sur C2 voire aucune.

Pour les ED, le problème du démarrage à froid est plus complexe que dans le contexte du Web car nous sommes confrontés à un nouveau système : nouvel item (un nouveau cube) et nouveaux utilisateurs (nouveaux décideurs ou décideurs connus avec une nouvelle fonction). Par conséquent, nous devons aller au-delà de la simple recommandation en suggérant des requêtes qui n'ont pas encore été exécutées sur le cube considéré. Dans le domaine des ED, à notre connaissance, il n'existe pas de recherche sur ce problème de démarrage à froid pour de nouveaux systèmes. L'article est organisé comme suit : la section 2 motive notre proposition par un exemple d'application. La section 3 présente le modèle conceptuel servant de support à notre proposition. La section 4 détaille notre processus de démarrage à froid et la section 5 propose un bilan et des perspectives à nos travaux.

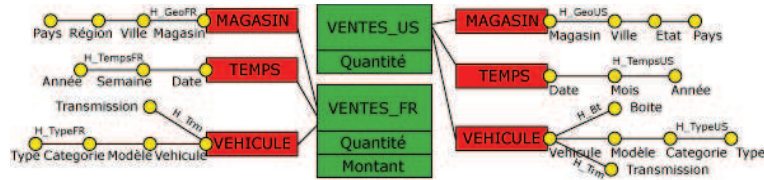
## 2. Exemple

### *Description détaillée de l'exemple*

Soit un décideur nouvellement employé dans l'industrie automobile. Il devra développer un réseau de revente dans un nouveau pays (Etats-Unis, US). La société a assemblé un nouvel entrepôt de données à partir de données de ventes de véhicules aux US disponibles publiquement et en le concevant de manière similaire à celui déjà existant dans l'entreprise (même fait, mêmes hiérarchies). L'ancien permet l'analyse des ventes de véhicules en France durant la dernière décennie, tandis que le nouveau système concerne les ventes de voitures aux US, mais seulement sur les deux dernières années (cf. les schémas en étoile *VENTES\_FR* et *VENTES\_US*, Figure 2). Il est à noter que dans les deux schémas, toutes les hiérarchies des dimensions se terminent par un niveau *ALL* ( $All^{Magasin}$ ,  $All^{Temps}$  et  $All^{Vehicule}$ ) et non affiché. Néanmoins, les différences suivantes existent : Aux US, la quantité de véhicules vendus est enregistré, en France, on trouve également le montant des ventes ; Les villes sont regroupées en régions en France et en états aux US. Les ventes sont regroupées par semaines en France et mensuellement aux US. Les véhicules vendus en France ont une boîte de vitesse manuelle, mais aux US elle peut différer (manuelle, automatique). En outre des différences existent au niveau des données : les zones géographiques sont soit aux États-Unis, soit en France et la période couverte est 1998-2011 (France) ou 2009-2011 (US). Les types de véhicules sont identiques mais, les modèles sont différents et les catégories ne sont pas toutes identiques (par ex. les petites voitures urbaines françaises). La transmission, en France, est traction avant ou 4x4, et aux US, il existe également la propulsion arrière.

### *Log de sessions dans le schéma en étoile français*

Le système français a déjà été utilisé et un ensemble de log de session d'analyse en ont été extraits. Ces logs sont les manipulations successives (l'enchaînement de requêtes OLAP) des utilisateurs. Dans l'exemple, (cf. Table 1, haut), un utilisateur français commence par ( $q_3$ ) : analyse des quantités vendues en France



**Figure 1.** Entrepôts de données pour analyser les ventes de véhicules : (droite) aux Etats-Unis de 2009 à 2011 ; (gauche) en France de 1998 à 2008.

Log de sessions				
	Ventes_FR	Véhicule	Magasin	Temps
$q_3$	Quantité	$All^{Vehicule}$	France (Pays)	2007 (Année)
$q_3 \rightarrow q_4$	Identite	Identite	Drilldown	Identite
$q_4$	Quantité	$All^{Vehicule}$	Nord (Région)	2007 (Année)
$q_4 \rightarrow q_5$	Identite	Identite	Drilldown	Identite
$q_5$	Quantité	$All^{Vehicule}$	Villes du nord (Villes)	2007 (Année)
Session courante				
	Ventes_US	Véhicule	Magasin	Temps
$q_1$	Quantité	$All^{Vehicule}$	US (Pays)	2011 (Année)
$q_1 \rightarrow q_2$	Identite	Identite	Drilldown	Identite
$q_2$	Quantity	$All^{Vehicule}$	Texas (État)	2011 (Année)
$q_2 \rightarrow q_{pred}$	Identite	Identite	Drilldown	Identite
$q_{pred}$	Quantité	$All^{Vehicule}$	Villes du Texas (Ville)	2011 (Année)

**Tableau 1.** Haut : log d'une session, ancien système ; Bas : analyse en cours, nouveau système.

durant 2007 ; puis, en deux étapes, il fore vers le bas (DrillDown) depuis le niveau Pays vers Région ( $q_4$ ), puis jusqu'à Ville ( $q_5$ ).

#### Recommandations pour l'exploration du schéma en étoile US

Le système US est neuf et n'a pas encore de requêtes à recommander à partir de ses propres logs et l'utilisateur n'a ni expérience avec la vente de véhicules, ni avec les pratiques usuelles des autres analystes de l'entreprise. En outre, 1) les schémas des deux systèmes ne sont pas identiques, 2) les données sont différentes (France contre US). Ainsi, le nouveau décideur ne peut être aidé ni par le système existant ni par les autres utilisateurs. Une exploration guidée de l'entrepôt de données pour le nouveau décideur pourrait être suggérée, mais sans prendre en compte les (nouvelles) données. Par exemple, le nouvel utilisateur analyse les quantités vendues de véhicules aux États-Unis durant l'année 2011 ( $q_1$  en Table 1). S'il fore vers le bas (DrillDown) sur le magasin depuis le niveau Pays vers État ( $q_2$ ), le système pourrait suggérer une façon " française " d'exploration : pour analyser les quantités de produits vendus pour une année spécifique, les utilisateurs de France, font systématiquement un double forage vers le bas sur la hiérarchie géographique ( $q_{pred}$ ).

#### Recommandation d'opérations de haut niveau

Typiquement dans cet exemple, les suggestions d'analyse doivent être indépendantes des données (US : ventes de 2011 ; France : ventes de 2007). Les deux schémas sont légèrement différents, mais ils ont tous deux un fait " ventes ", une

dimension avec une " localisation géographique ", une dimension temporelle et une troisième représentant des produits vendus (des véhicules). Sans avoir des structures identiques, les dimensions sont assez similaires, ainsi l'exploration peut être similaire entre les deux schémas. Afin d'y parvenir, les recommandations devront prendre en compte uniquement des opérations de haut niveau (double forage sur une hiérarchie géographique ; rotation depuis les localisations vers les produits). Ces opérations reposent sur les opérateurs algébriques OLAP.

Ainsi, des recommandations basées sur des opérations plutôt que des valeurs (des données) peuvent être utiles pour le problème du démarrage à froid des systèmes de recommandations.

### 3. Modélisation Conceptuelle

Cette section définit le modèle multidimensionnel, extension de travaux antérieurs [Ravat *et al.*2008]. L'approche s'inscrit à un niveau d'abstraction conceptuelle afin de définir des concepts indépendamment des contraintes d'implantation.

#### 3.1. Schéma Multidimensionnel et cube

On pose :  $F = \{F_1, \dots, F_n\}$  un ensemble fini de faits,  $n \geq 1$  et  $D = \{D_1, \dots, D_m\}$  un ensemble fini de dimensions,  $m \geq 2$ .

**Définition 1.** Un schéma de cube, noté  $C_i$ , est défini par  $(F_i, Star(F_i))$ , tel que :  $F_i \in F$  est un fait et  $Star(F_i) = \{D^{n^{D_j}} \mid \forall j \in 1..m, D^{n^{D_j}} \in D\}$  est l'ensemble des dimensions associées au fait  $F_i$ .

Un schéma de cube décrit les données en fonction de sujets d'analyses (faits), et d'axes d'analyses (dimensions). Les dimensions sont hiérarchisées pour décrire les différents niveaux de granularité des données.

Soient  $\mathcal{N} = \{n_1, n_2, \dots\}$  un ensemble fini de noms non redondants.

**Définition 2.** Un fait, noté  $\forall i \in 1..n ]F_i$ , est défini par  $(n^{F_i}, M^{F_i})$ , tel que :

- $n^{F_i} \in \mathcal{N}$  est le nom identifiant le fait,
- $M^{F_i} = \{m_1, \dots, m_{p_i}\}$  est un ensemble de mesures.

**Définition 3.** Une dimension, notée  $\forall i \in [1..m]D_i$ , est définie par  $(n^{D_i}, A^{D_i}, H^{D_i})$ , tel que :

- $n^{D_i} \in \mathcal{N}$  est le nom identifiant la dimension ,
- $A^{D_i} = \{a_1^{D_i}, \dots, a_{r_i}^{D_i}\}$  est l'ensemble des attributs de la dimension ,
- $H^{D_i} = \{H_1^{D_i}, \dots, H_{s_i}^{D_i}\}$  est un ensemble de hiérarchies.

Les hiérarchies organisent les attributs d'une dimension, appelés paramètres, de la graduation la plus fine, notée  $Id^{D_i}$ , jusqu'à la graduation la plus générale, notée  $All^{D_i}$ . Une hiérarchie définit les chemins de navigation valides sur un axe d'analyse en décrivant les différents niveaux de granularités possibles auxquels les mesures du fait sont observables.

**Définition 4.** Une hiérarchie, notée  $H_j$  (notation abusive de  $H_j^{D_i}, \forall i \in [1..m], \forall j \in [1..s_i]$ ) est définie par  $(n^{H_j}, P^{H_j}, \prec^{H_j})$ , tel que :

- $n^{H_j} \in \mathcal{N}$  est le nom identifiant la hiérarchie ,
- $P^{H_j} = \{p_1^{H_j}, \dots, p_{q_j}^{H_j}\}$  est un ensemble d'attributs de la dimension appelés paramètres,  $P^{H_j} \subseteq A^{D_i}$ ,
- $\prec^{H_j} = \{(p_x^{H_j}, p_y^{H_j}) \mid p_x^{H_j} \in P^{H_j} \wedge p_y^{H_j} \in P^{H_j}\}$  est une relation binaire antisymétrique et transitive. L'antisymétrie signifie que  $(p_{k_1}^{H_j} \prec^{H_j} p_{k_2}^{H_j}) \wedge (p_{k_2}^{H_j} \prec^{H_j} p_{k_1}^{H_j}) \Rightarrow p_{k_1}^{H_j} = p_{k_2}^{H_j}$  et la transitivité signifie que  $(p_{k_1}^{H_j} \prec^{H_j} p_{k_2}^{H_j}) \wedge (p_{k_2}^{H_j} \prec^{H_j} p_{k_3}^{H_j}) \Rightarrow p_{k_1}^{H_j} \prec^{H_j} p_{k_3}^{H_j}$ .
- $Weak^{H_j} : P^{H_j} \rightarrow 2^{A^{D_i} \setminus P^{H_j}}$  est une application qui associe à chaque paramètre un ensemble d'attributs de dimension, appelés attributs faibles.

On pose  $P = \bigcup_{i=1}^m A^{D_i} = \bigcup_{i=1}^m \bigcup_{j=1}^{S_i} P^{H_j}$

Par abus de notation dans la suite de l'article, nous décrirons les hiérarchies conformément à la notation simplifiée suivante  $H_j = (n^{H_j}, P^{H_j}, \prec^{H_j}) = (n^{H_j}, path^{H_j})$  où  $path^{H_j} = \langle Id^{D_i}, \dots, All^{D_i} \rangle$  est un ensemble ordonné de paramètres du paramètre racine  $Id^{D_i}$  au paramètre extrémité  $All^{D_i}$ .

*Exemple*

La figure 1 présente un exemple de schéma de cube  $(F_1, Star(F_1))$  correspondant à l'analyse des ventes de véhicules entre 2009 et 2011. Ce cube est conforme à un schéma multidimensionnel représenté suivant des notations graphiques [Ravat et al.2007], [Ravat et al.2008]. Il est défini formellement comme suit.

- $F_1 = ('VENTE-US', Quantite),$
- $Star(F_1) = \{D^{MAGASIN}, D^{TEMPS}, D^{VEHICULE}\},$  où
  - $D^{MAGASIN} = ('Magasin', \{Magasin, Ville, Etat, Pays\}, \{('H_GeoUS', \langle Magasin, Ville, Etat, Pays \rangle)\}),$
  - $D^{TEMPS} = ('Temps', \{Date, Mois, Annee\}, \{('H_TpsUS', \langle Date, Mois, Annee \rangle)\}),$
  - $D^{VEHICULE} = ('Vehicule', \{Vehicule, Modele, Categorie, Marque, Transmission, Moteur\}, \{('H_Tr', \langle Vehicule, Transmission \rangle), ('H_MqUS', \langle Vehicule, Modele, Categorie, Marque \rangle), ('H_Mt', \langle Vehicule, Moteur \rangle)\}).$

### 3.2. Analyses OLAP

Les systèmes OLAP favorisent l'analyse interactive des données en offrant un ensemble d'opérations spécifiques telles que les forages (drill-down, roll-up), les sélections (slice-and-dice) [Ravat et al.2007]. La navigation OLAP s'apparente à l'application d'une succession d'opérations : l'utilisateur initialise sa

$Display(F_{NEW}, f_i(M_{NEW}), \{D_{SHOW}, H_{SHOW}, P_{SHOW}\}) = q_{RES}$ Affichage du fait $F_{NEW}$ et de sa mesure $M_{NEW}$ agrégée par $f_i$ en fonction des dimensions $\{D_{SHOW}\}$ . Le paramètre utilisé par défaut est le paramètre extrémité $All^{D_i}$ .
$Pivot(q_{SRC}, D_{CUR}, D_{NEW}, H_{NEW}, P_{NEW}) = q_{RES}$ Changement de l'axe d'analyse $D_{CUR}$ par un nouvel axe $D_{NEW}$ sur sa hiérarchie $H_{NEW}$ . Le paramètre utilisé par défaut est le paramètre extrémité $All^{D_{NEW}}$ .
$DrillDown(q_{SRC}, D_{CUR}, P_{NEW}) = q_{RES}$ Affichage des données en introduisant un nouveau paramètre $P_{NEW}$ de plus fine granularité.
$RollUp(q_{SRC}, D_{CUR}, P_{CUR}) = q_{RES}$ Affichage des données en utilisant un paramètre $P_{CUR}$ de plus haute granularité (suppression des paramètres hiérarchiquement inférieur à $P_{CUR}$ ).

**Tableau 2.** Algèbre OLAP.

navigation (ou analyse) par une première requête permettant d'obtenir un résultat qui est ensuite transformé par une succession d'opérations jusqu'à obtenir le résultat pertinent pour l'utilisateur. Ce dernier explore ainsi l'espace multidimensionnel de l'entrepôt de données.

Une visualisation classique consiste à voir les données sous une représentation en cube, affichant un fait et les informations détaillées de toutes les dimensions liées. Un cube présente simultanément les structures et les valeurs des données manipulées. Nous désignons par squelette de requête les structures d'un cube de données.

**Définition 5.** Un squelette de requête  $P_{q_i}$  est défini par  $(M_k, Level)$ , où :

- $M_k \in M^{F_i}$  est une mesure du fait  $F_i$ ,
- $Level : Star(F_i) \rightarrow P$  est une fonction qui associe chaque dimension à un ensemble de paramètres.

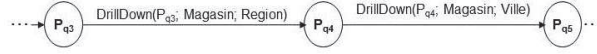
#### Exemple

La figure 5 montre un exemple de squelettes de requêtes qui se définissent formellement comme suit.

$$\begin{aligned}
P_{q_3} &: (\{Quantite\}, \{level(Vehicule) = All^{Vehicule}, level(Magasin) = Pays, level(Tps) = Annee\}) \\
P_{q_4} &: (\{Quantite\}, \{level(Vehicule) = All^{Vehicule}, level(Magasin) = Region, level(Tps) = Annee\}) \\
P_{q_5} &: (\{Quantite\}, \{level(Vehicule) = All^{Vehicule}, level(Magasin) = Ville, level(Tps) = Annee\})
\end{aligned}$$

Nous définissons ainsi l'analyse d'un utilisateur par un graphe où chaque arc représente une opération OLAP et chaque noeud représente le résultat de l'opération. Chaque noeud est défini par un squelette de requête décrivant la structure de la table multidimensionnelle résultat. Chaque arc est défini par une opération OLAP. Il faut noter qu'il n'existe pas de consensus sur une algèbre OLAP de référence à l'image de ce qui existe dans le domaine des bases de données relationnelles. Le tableau 2 décrit l'ensemble des opérateurs OLAP que nous prenons en compte dans notre approche. Nous limitons cet ensemble à un noyau d'opérateurs permettant de transformer la structure des tables multidimensionnelles.





**Figure 2.** Extrait d'un exemple de graphe de squelette de session.

L'opérateur DISPLAY définit les sessions. Une session débute par l'opérateur DISPLAY et se termine par un nouvel opérateur DISPLAY.

**Définition 6.** *Un squelette de session est définie par un graphe  $(V, E)$  tel que :*

- $V = \{P_{q_1}, P_{q_2}, P_{q_3} \dots\}$  est un ensemble de squelettes de requêtes,
- $E = \{(Op_k, P_{q_{k+1}}) \mid Op_k \text{ est un opérateur OLAP et } P_{q_{k+1}} \in V\}$

#### Exemple

La figure 3 montre l'extrait d'un squelette de session défini formellement comme suit.

- $V = \{\dots P_{q_3}, P_{q_4}, P_{q_5} \dots\},$
- $E = \{\dots (DrillDown(P_{q_3}; Magasin; Region), P_{q_4}), (DrillDown(P_{q_4}; Magasin; Ville), P_{q_5}) \dots\}$

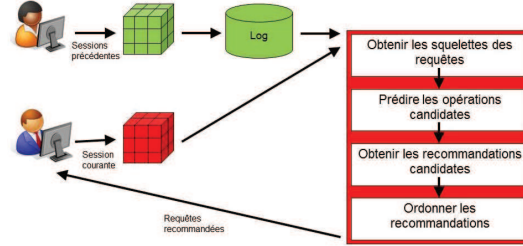
## 4. Processus de démarrage à froid

Dans cette section, nous détaillons le processus de démarrage à froid dans le cadre de notre système de recommandation de requêtes OLAP. Ce processus utilise la séquence de requêtes de la session courante ainsi que le log de requêtes du serveur OLAP. Ce log contient les séquences de requêtes précédemment lancées sur un autre cube qui est similaire au cube sur lequel sont lancées les requêtes de la session courante. Nous proposons d'utiliser les travaux de [Sbouï *et al.*2010] sur l'interopérabilité des cubes afin de détecter des cubes similaires. Cependant, la notion de similarité entre deux cubes de données sera définie dans nos travaux futurs.

Notre processus (cf. figure 4) inclut les quatre étapes suivantes : 1) Obtenir les squelettes des requêtes du log ; 2) Prédire les opérations candidates en utilisant le squelette de la session courante, l'ensemble des squelettes des sessions du log, et une fonction de concordance "match" entre le squelette de la session courante et l'ensemble des squelettes des sessions ; 3) Calculer les recommandations candidates en combinant une requête de la session courante et les opérations candidates et 4) Ordonner les recommandations candidates. Chaque étape est détaillée par la suite.

### 4.1. Obtenir les squelettes des requêtes du log

Initialement, chaque session du log est transformée en squelettes de requêtes. A cet effet, cette étape utilise une fonction permettant d'obtenir le squelette de chaque requête (voir l'algorithme 1). Cette fonction retourne un ensemble d'ensembles de squelettes de requêtes, que nous appelons un ensemble de squelettes



**Figure 3.** Vue d'ensemble du processus de démarrage à froid.

de sessions. Le principe est le suivant : Pour chaque session du log, il s'agit de remplacer dans la session, chaque requête par son squelette en utilisant la fonction `extractReqSquel` (voir l'algorithme 2) et de détailler les opérations qui permettent de passer d'une requête à sa suivante dans la session (voir la fonction `extractOp`, algorithme 3). Le squelette de la session courante est obtenu de la même manière.

---

#### Algorithm 1 `LogSquel(L, CL)`

---

**Entrée:**

$L$  : Le log de requêtes précédemment lancées, i.e. un ensemble de sessions,

$C_L$  : Le cube sur lequel les requêtes du log ont été lancées.

**Sortie:** un ensemble de squelettes de sessions ( $SP$ )

---

$V \leftarrow \emptyset$  // pour les opérations

$E \leftarrow \emptyset$  // pour les squelettes de requêtes

$SP \leftarrow \langle E, V \rangle$

**Pour** chaque session  $s_i \in L$  **Faire**

**Pour** chaque couple de requêtes  $\langle q_j, q_{j+1} \rangle \in s_i$  **Faire**

$SP.E \leftarrow SP.E \cup \text{extractReqSquel}(q_j, C_L)$

        // `extractReqSquel` : une fonction qui extrait le squelette d'une requête donnée.

$SP.E \leftarrow SP.E \cup \text{extractReqSquel}(q_{j+1}, C_L)$

$SP.V \leftarrow SP.V \cup \text{extractOp}(q_j, q_{j+1}, C_L)$

        // `extractOp` : Une fonction qui extrait l'opération permettant de passer d'une requête à l'autre dans une session donnée.

**Fin Pour**

**Fin Pour**

**Retourne**  $SP$

---

L'algorithme 2 montre comment les squelettes de requêtes peuvent être obtenus tandis que l'algorithme 3 détaille comment sont identifiées les opérations permettant de passer d'une requête à une autre.

L'obtention des squelettes de requêtes (voir l'algorithme 2) consiste à extraire les références (ensemble des valeurs des attributs) d'une requête donnée et de retourner, pour chaque dimension, le nom du niveau correspondant.

---

#### Algorithm 2 `extractReqSquel(q, C)`

---

**Entrée:**

$q$  : Une requête donnée,

$C$  : Le cube sur lequel la requête  $q$  a été lancée.

**Sortie:** une liste de noms de niveaux du cube ( $QP$ )

---

$QP \leftarrow \emptyset$

**Pour** chaque dimension  $D_i$  de  $C$  **Faire**

$QP \leftarrow QP \cup \text{getNomNiveau}(\text{getReferences}(q, C))$

**Fin Pour**

**Retourne**  $QP$

---

L'extraction des opérations (algorithme 3) consiste à extraire les références (un ensemble de valeurs d'attributs) de deux requêtes données, les numéros de niveaux correspondants sur chaque dimension et à trouver l'opération qui permet de naviguer d'une requête à l'autre. Nous nous intéressons en particulier aux opérations OLAP classiques : *slice-and-dice*, *drill-down*, *roll-up*, *switch* et *pivot*. Il est à noter que les opérations *slice-and-dice* et *switch* n'ont pas d'influence sur notre approche puisque ces opérations impactent uniquement les valeurs. Notre approche intervient à un niveau plus élevé, elle manipule la structure du cube et non ses valeurs. Plus précisément, pour deux requêtes données  $(q_1, q_2)$  et pour chaque dimension du cube  $C$  : si les références (ensembles d'attributs) de chaque requête sont localisées sur le même niveau hiérarchique de la dimension, l'opération pour passer d'une requête à l'autre est *l'identité* ; si les références d'une seule des requêtes sont localisées au niveau le plus élevé de la hiérarchie (niveau '*ALL*'), l'opération est un *pivot* ; si les références de la première requête lancée sont localisées à un niveau plus élevé que les références de la seconde requête lancée, l'opération est un forage vers le bas, *DrillDown* ; dans tous les autres cas, l'opération est un forage vers le haut, *RollUp*. Lorsque le système détecte un *pivot*, il s'agit de déterminer sur quelle dimension le *pivot* a lieu.

---

### Algorithm 3 extractOp( $q_1, q_2, C$ )

---

**Entrée:**

$q_1, q_2$  : Deux requêtes données,

$C$  : Le cube sur lequel les requêtes  $q_1$  et  $q_2$  ont été lancées.

**Sortie:** une liste d'opérations OLAP ( $OP$ )

---

```

 $OP, Pivot \leftarrow \emptyset$ 
 $Niv_1, Niv_2, hauteur \leftarrow 0$ 
Pour chaque dimension  $D_i$  de  $C$  Faire
     $Niv_1 \leftarrow getNumNiveau(getReferences(q_1, C))$ 
     $Niv_2 \leftarrow getNumNiveau(getReferences(q_2, C))$ 
     $hauteur \leftarrow$  hauteur de la hiérarchie de la dimension  $D_i$ 
    Si  $Niv_1 - Niv_2 == 0$  Alors
         $OP \leftarrow OP \cup "Identite"$ 
    Fin Si
    Si  $(Niv_1 \neq hauteur \wedge Niv_2 == hauteur) \vee (Niv_1 == hauteur \wedge Niv_2 \neq hauteur)$  Alors
         $Pivot \leftarrow Pivot \cup D_i$ 
    Fin Si
    Si  $Niv_1 - Niv_2 > 0$  Alors
         $OP \leftarrow OP \cup "Drilldown"$ 
    Fin Si
    Si  $Niv_1 - Niv_2 < 0$  Alors
         $OP \leftarrow OP \cup "RollUp"$ 
    Fin Si
Fin Pour
Si  $|Pivot| > 1$  Alors
    Pour chaque couple de dimensions  $\langle D_i, D_{i+1} \rangle$  du  $Pivot$  Faire
         $OP[D_i] \leftarrow "Pivot(" + D_i + ", " + D_{i+1} + ")"$ 
         $OP[D_{i+1}] \leftarrow "Pivot(" + D_i + ", " + D_{i+1} + ")"$ 
    Fin Pour
Fin Si
Retourne  $OP$ 

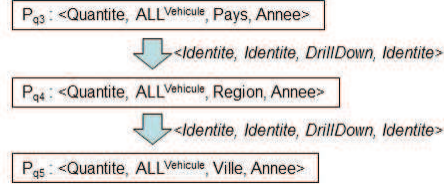
```

---

La fonction  $getReferences(q, C)$  retourne pour chaque dimension de  $C$ , les valeurs des attributs de la requête  $q$ . Par exemple<sup>1</sup>, les références de la requête  $q_1$  de l'exemple de la section 2 sont  $\langle Quantite, All^{Vehicule}, US, 2011 \rangle$ . La fonc-

---

1. Il faut noter que, par souci de lisibilité, les références de requêtes, les squelettes des requêtes et les opérations sont présentés, dans cette section, comme des 4-uplets ordonnés où chaque tuple correspond, dans cet ordre, aux mesures, puis à la dimension



**Figure 4.** *Squelette de la session de l'exemple de motivation.*

tion *getNomNiveau* (resp. *getnumNiveau*) extrait, pour chaque ensemble de valeurs d'attributs de chaque dimension du cube  $C$ , le nom (resp. le rang dans la hiérarchie - 0 pour la plus fine granularité) de chaque niveau correspondant dans la hiérarchie de la dimension. Pour  $q_1$ , nous obtenons *Annee* comme nom de niveau pour la valeur d'attribut 2011 et 2 qui est le numéro du niveau de *Annee* dans la hiérarchie TEMPS de  $C$  (pour *Mois* ce serait 1 et 0 pour *Date*).

Par exemple, dans le cas de la requête  $q_3$ , qui a pour références  $\langle Quantite, All^{Vehicule}, France, 2007 \rangle$  (voir Tableau 1), son squelette peut être :  $\langle Quantite, All^{Vehicule}, Pays, Annee \rangle$ . Ainsi, la session de notre exemple, qui contient  $q_3$ ,  $q_4$  et  $q_5$ , peut avoir pour squelette celui présenté sur la figure 3.

---

**Algorithm 4** PredictOpCandidates( $s_c, C, SP, Match$ )

---

**Entrée:**  
 $s_c$  : La session courante,  
 $C$  : Le cube sur lequel la session  $s_c$  a été lancée,  
 $SP$  : l'ensemble des squelettes de sessions, i.e., le résultat de l'étape précédente,  
 $Match$  : Une fonction qui retourne les squelettes de sessions candidates.  
**Sortie:** un ensemble d'opérations candidates ( $Cand$ )

---

```

M, Cand ← ∅
Psc ← le squelette de la session courante sc
M ← Match(Psc, SP)
Si M ≠ ∅ Alors
  Pour chaque m = ⟨p, pos⟩ ∈ M Faire
    //où p = ⟨e, v⟩ est un squelette de session
    Cand ← Cand ∪ p.v(p.e[pos], p.e[pos + 1])
  Fin Pour
Fin Si
Retourne Cand
  
```

---

#### 4.2. Prédire les opérations candidates

L'étape précédente produit l'ensemble des squelettes des sessions. Maintenant, en utilisant cet ensemble et la séquence de requêtes de la session courante, l'algorithme 4 permet de construire un ensemble d'opérations candidates : En

---

Véhicule, puis à la dimension Magasin et enfin à la dimension Temps. Par exemple, la référence de requête  $\langle m, a, b, c \rangle$  indique que cette requête traite de la mesure  $m$ , en fonction des valeurs d'attributs  $a$  sur la dimension Véhicule,  $b$  sur la dimension Magasin et  $c$  sur la dimension Temps. Le squelette de requête  $\langle m, pa, pb, pc \rangle$  indique que ce squelette traite de la mesure  $m$  aux niveau  $pa$  sur la dimension Véhicule,  $pb$  sur la dimension Magasin et  $pc$  sur la dimension Temps. Et l'opération  $\langle o_1, o_2, o_3, o_4 \rangle$  indique qu'il y a les opérations  $o_1$  sur les mesures,  $o_2$  sur la dimension Véhicule,  $o_3$  sur la dimension Magasin et  $o_4$  sur la dimension Temps.

plus de l'ensemble des squelettes des sessions et de la session courante, cet algorithme utilise une fonction de concordance (*matching* : Match). Cette fonction est utilisée pour trouver l'ensemble des squelettes des sessions du log qui coïncident avec le squelette de la session courante. L'algorithme procède de la manière suivante : Dans un premier temps, le squelette de la session courante est construit (la fonction LogSql est utilisée sur la session courante qui peut être vue comme un log contenant une seule session). Puis, la fonction Match est utilisée pour chercher, parmi l'ensemble des squelettes de sessions, celui qui coïncide avec le squelette de la session courante. Cette fonction retourne un ensemble de paires de sessions indiquant quels squelettes de sessions coïncident avec le squelette de la session courante ainsi que la position de la concordance. A partir de ces paires, l'algorithme extrait un ensemble d'opérations candidates qui sont les opérations qui permettent de passer du squelette de requête, situé à la position retournée dans le squelette de session qui coïncide, au squelette de requête suivant dans le squelette de la session candidate. Cet ensemble d'opérations est retourné comme résultat. Il faut noter que notre objectif est d'éviter que cet ensemble soit vide et qu'un tel algorithme a une complexité de  $O(n[(w+1)^2 + 1] + w)$  où  $n$  est le nombre de sessions du log et  $w$  est la longueur maximale (nombre de requêtes) d'une session donnée.

Par exemple, le squelette de la session  $s_1$  de l'exemple de la section 2 qui contient  $q_3$ ,  $q_4$  et  $q_5$  est représentée sur la figure 3. Le squelette de la session courante  $s_c$  contient le squelette de  $q_1$  :  $\langle Quantite, All^{Vehicule}, Pays, Annee \rangle$ , le squelette de  $q_2$  :  $\langle Quantite, All^{Vehicule}, Etat, Annee \rangle$  et les opérations qui transforment  $q_1$  en  $q_2$  :  $\langle Identite, Identite, Drilldown, Identite \rangle$ . Par exemple, supposons que la fonction Match retourne le squelette de  $s_1$  :  $P_{s_1}$  qui coïncide avec le squelette de  $s_c$  :  $P_{s_c}$  à la position 2. L'algorithme 4 retourne alors  $\langle Identite, Identite, Drilldown, Identite \rangle$  en tant qu'opération candidate. En effet, cette opération de forage vers le bas (*drill-down*) transforme la requête  $q_4$  à la position 2 dans la session  $s_1$  en la requête  $q_5$  à la position 3 dans  $s_1$ .

#### *La fonction Match*

Cette fonction Match a été étudiée dans [Negre2009] où des fonctions de concordance de séquences sont détaillées (chaque fonction est basée sur une distance d'édition et sur une concordance approximative de chaînes de caractères (*approximate string matching*)) et combinées avec des mesures de similarité entre membres (le plus court chemin et la distance de Hamming).

La technique de concordance approximative de chaînes utilisée consiste à supprimer, à chaque itération, le dernier élément de la séquence puis de comparer la sous-séquence obtenue à la séquence courante en utilisant une distance d'édition. Il faut noter que le nombre de calculs de la distance d'édition peut être exponentiel. Alors que, si, seule la distance d'édition est calculée (i.e. la distance de Levenshtein), le nombre de calculs est plus petit, puisqu'elle est calculée une seule fois pour chaque séquence. Ainsi, la distance de Levenshtein a été utilisée pour comparer deux sessions. Cette distance est combinée à la

distance de Hamming ( $EdH$ ) ou au plus court chemin ( $EdSP$ ) pour comparer des membres. Les expérimentations menées dans [Negre2009] montrent que  $EdH$  et  $EdSP$  ont des performances similaires en termes de rappel/précision mais que  $EdSP$  est moins rapide que  $EdH$ . Comme nous nous situons dans le contexte du problème de démarrage à froid d'un système de recommandation, le système proposé doit être le plus rapide et le plus efficace (autant que possible). C'est la raison pour laquelle, nous avons décidé d'utiliser  $EdH$  : la distance de Levenshtein combinée avec la distance de Hamming.

#### 4.3. Calculer les recommandations candidates

L'étape précédente produit un ensemble d'opérations candidates. Suite à cela, pour chaque opération candidate, une nouvelle requête est construite en appliquant ces opérations candidates à la dernière requête de la session courante. Une algèbre multidimensionnelle proposée récemment [Romero *et al.*2011] pourrait garantir que les opérations appliquées auront un sens.

Dans [Negre2009], cinq possibilités ont été considérées : (i) la dernière requête de la session courante, (ii) le successeur d'une requête donnée de la session courante, (iii) l'union des requêtes de la session courante, (iv) l'intersection des requêtes de la session courante, et (v) le médoïde<sup>2</sup> des requêtes de la session courante. La possibilité (v) est chronophage lorsque les sessions sont longues (à cause du calcul du médoïde pour un grand nombre de requêtes). La possibilité (iv) peut aboutir à un ensemble vide de requêtes et la possibilité (iii) peut créer une requête dont le résultat peut correspondre à l'intégralité du cube de données. Par analogie avec le web [White2007] et vis-à-vis de notre contexte de démarrage à froid, nous supposons que des étapes intermédiaires sont inutiles et qu'une session d'analyse se concentre sur le but de la session. Ainsi, pour nous, la meilleure possibilité est d'appliquer les opérations à la dernière requête de la session courante afin d'obtenir les recommandations candidates.

Dans notre exemple, il y a une opération candidate :  $\langle Identite, Identite, Drilldown, Identite \rangle$ . La construction d'une nouvelle requête en appliquant cette opération (un *drill-down* sur la dimension Magasin) sur la dernière requête  $q_2$  (les quantités de véhicules vendus au Texas en 2011) de la session courante  $s_c$  retourne la requête identifiée dans la table 1 par  $q_{pred}$  (les quantités de véhicules vendus dans les villes du Texas en 2011).

#### 4.4. Ordonner les recommandations candidates

A partir de l'ensemble de recommandations obtenu précédemment, l'idée est de sélectionner les recommandations les plus pertinentes. Nous considérons que nous ne pouvons pas avoir de critère de satisfaction exprimé par l'utilisateur. Par conséquent, un ordonnancement de requêtes, qui ordonne les recommandations candidates, est difficile à proposer.

---

2. Le médoïde d'un ensemble de requêtes appartient à cet ensemble et est la requête qui minimise les distances avec les autres requêtes de l'ensemble.

A l'heure actuelle, les solutions que nous pouvons considérées sont : (i) ordonner les candidats en fonction de leur proximité avec la dernière requête de la session courante, (ii) ordonner les candidats en fonction de leur nombre d'occurrences dans les logs, (iii) ordonner les candidats en fonction de la position de l'opération candidate dans le squelette du log, i.e. les opérations les plus récentes lancées dans le log seront utilisées pour construire les premières requêtes de l'ensemble des requêtes recommandées.

Tout d'abord, nous ne connaissons ni la densité du log ni son contenu. Ainsi, ordonner les candidats en fonction de leur nombre d'occurrences dans les logs peut être difficile, si, par exemple, chaque candidat apparait le même nombre de fois. Toujours pour des raisons de rapidité, ordonner les candidats en fonction de leur proximité avec la dernière requête de la session courante peut être chronophage. Pour ces raisons, les candidats sont donc ordonnés en fonction de la position de l'opération candidate dans le squelette du log.

Finalement, notre proposition de démarrage à froid pour un système de recommandations, qui sera implémentée dans nos travaux futurs, consiste à :

- 1) Obtenir les squelettes des requêtes du log (précédemment lancées sur un cube similaire) en utilisant notre algorithme *LogSquel*.
- 2) Prédire les opérations candidates en utilisant le squelette de la session courante et l'ensemble des squelettes des sessions du log et la fonction de concordance *EdH* entre le squelette de la session courante et l'ensemble des squelettes des sessions, dans notre algorithme *PredictOpCandidates*
- 3) Obtenir les recommandations candidates en combinant la dernière requête de la session courante et les opérations candidates.
- 4) Ordonner les requêtes candidates en fonction de la position de l'opération candidate dans le squelette du log.

## 5. Conclusion et discussions

Dans cet article, nous avons proposé un système de recommandations lors de l'exploration de nouveaux cubes de données. Notre processus de prédictions de requêtes est basé sur l'analyse du comportement du décideur durant ses sessions d'analyse (séquence de requêtes), et, sur le comportement de l'utilisateur au cours de sessions d'analyses passées sur un autre système.

Cette approche prédictive soulève quatre défis : (1) extraire le comportement des utilisateurs en définissant le squelette de requêtes en cours et en analysant les journaux de requêtes sur d'anciens cubes de données (2) prévoir des opérations candidates de manipulation OLAP via la correspondance de squelettes de requêtes, (3) sélectionner les opérations pertinentes parmi l'ensemble précédemment défini, (4) classer ces recommandations candidates. Ce système de recommandation basé sur le principe du démarrage à froid sera utilisé jusqu'à ce

que le système recueille suffisamment de données utilisateur sur leurs analyses décisionnelles afin de pouvoir utiliser un système de recommandation standard.

A cours terme, nous avons l'intention de réaliser des expérimentations et d'étendre ces travaux à un cadre d'application concret. Cela nous permettra de tester nos algorithmes dans un contexte précis et de répondre aux questions suivantes : comment identifier le meilleur squelette de requête si nous en avons plusieurs ? Comment les ordonner lorsque le système propose un trop grand nombre suggestions ? De plus, nous avons l'intention de prendre en compte d'autres opérateurs OLAP plus complexes tels que NEST et Drill-Across. Enfin, nous souhaitons également appliquer nos algorithmes sur des schémas multidimensionnels possédant des différences plus importantes que ceux étudiés dans cet article (autrement dit, de nouveaux schémas composés de dimensions ou de faits différents).

## 6. Bibliographie

- [Adomavicius et Tuzhilin2005] Adomavicius G., Tuzhilin A., « Toward the Next Generation of Recommender Systems : A Survey of the State-of-the-Art and Possible Extensions », *IEEE Trans. Knowl. Data Eng.*, vol. 17, n° 6, 2005, p. 734-749.
- [Baeza-Yates2010] Baeza-Yates R. A., « Query intent prediction and recommendation », *RecSys*, 2010, p. 5-6.
- [Bellatreche et al.2005] Bellatreche L., Giacometti A., Marcel P., Mouloudi H., Laurent D., « A personalization framework for OLAP queries », *DOLAP*, 2005, p. 9-18.
- [Cariou et al.2008] Cariou V., Cubillé J., Derquenne C., Goutier S., Guisnel F., Klajnmic H., « Built-In Indicators to Discover Interesting Drill Paths in a Cube », *DaWaK*, 2008, p. 33-44.
- [Chatzopoulou et al.2009] Chatzopoulou G., Eirinaki M., Polyzotis N., « Query Recommendations for Interactive Database Exploration », *SSDBM*, 2009, p. 3-18.
- [Colliat1996] Colliat G., « OLAP, Relational, and Multidimensional Database Systems », *SIGMOD Record*, vol. 25, n° 3, 1996, p. 64-69.
- [Giacometti et al.2009] Giacometti A., Marcel P., Negre E., « Recommending Multidimensional Queries », *DaWaK*, 2009, p. 453-466.
- [Giacometti et al.2011] Giacometti A., Marcel P., Negre E., Soulet A., « Query Recommendations for OLAP Discovery-Driven Analysis », *IJDWM*, vol. 7, n° 2, 2011, p. 1-25.
- [Golbandi et al.2011] Golbandi N., Koren Y., Lempel R., « Adaptive bootstrapping of recommender systems using decision trees », *WSDM*, 2011, p. 595-604.
- [Golfarelli et al.2011] Golfarelli M., Rizzi S., Biondi P., « myOLAP : An Approach to Express and Evaluate OLAP Preferences », *IEEE Trans. Knowl. Data Eng.*, vol. 23, n° 7, 2011, p. 1050-1064.
- [Jerbi et al.2009] Jerbi H., Ravat F., Teste O., Zurfluh G., « Preference-Based Recommendations for OLAP Analysis », *DaWaK*, 2009, p. 467-478.
- [Khoussainova et al.2009] Khoussainova N., Balazinska M., Gatterbauer W., Kwon Y., Suciu D., « A Case for A Collaborative Query Management System », *CIDR*,



2009.

- [Kimball1996] Kimball R., *The Data Warehouse Toolkit : Practical Techniques for Building Dimensional Data Warehouses*, John Wiley, 1996.
- [Kłopotek2008] Kłopotek M. A., « Approaches to cold start in Recommender Systems », Kłopotek M. A., Tchórzewski J., Eds., *Proceedings of 10th International Conference on Artificial Intelligence*, vol. 5 de *Proceedings of Artificial Intelligence Studies*, Siedlce, Poland, septembre 18-19, 2008, p. 29–33.
- [Lyman et al.2003] Lyman P., Varian H. R., Charles P., Good N., Jordan L. L., Pal J., « How much information? », Available at <http://www2.sims.berkeley.edu/research/projects/show-much-info-2003>, 2003.
- [Marcel et Negre2011] Marcel P., Negre E., « A survey of query recommendation techniques for datawarehouse exploration », *EDA*, 2011.
- [Negre2009] Negre E., « Exploration collaborative de cubes de données », PhD thesis, Université François-Rabelais de Tours, 2009.
- [Rashid et al.2008] Rashid A. M., Karypis G., Riedl J., « Learning preferences of new users in recommender systems : an information theoretic approach », *SIGKDD Explorations*, vol. 10, n° 2, 2008, p. 90-100.
- [Ravat et al.2007] Ravat F., Teste O., Tournier R., Zurfluh G., « Graphical Querying of Multidimensional Databases », *ADBIS*, 2007, p. 298-313.
- [Ravat et al.2008] Ravat F., Teste O., Tournier R., Zurfluh G., « Algebraic and Graphic Languages for OLAP Manipulations », *IJDWM*, vol. 4, n° 1, 2008, p. 17-46.
- [Romero et al.2011] Romero O., Marcel P., Abelló A., Peralta V., Bellatreche L., « Describing analytical sessions using a multidimensional algebra », *Proceedings of the 13th international conference on Data warehousing and knowledge discovery, DaWaK'11*, Berlin, Heidelberg, 2011, Springer-Verlag, p. 224–239.
- [Sapia1999] Sapia C., « On Modeling and Predicting Query Behavior in OLAP Systems », *DMDW*, 1999, page 2.
- [Sarawagi2000] Sarawagi S., « User-Adaptive Exploration of Multidimensional Data », *VLDB*, 2000, p. 307-316.
- [Sbouï et al.2010] Sboui T., Salehi M., Bédard Y., « A Systematic Approach for Managing the Risk Related to Semantic Interoperability between Geospatial Databases », *IJAIS*, vol. 1, n° 2, 2010, p. 20-41.
- [Schein et al.2001] Schein A. I., Popescul A., Ungar L. H., Pennock D. M., « Generative Models for Cold-Start Recommendations », *IN PROCEEDINGS OF THE 2001 SIGIR WORKSHOP ON RECOMMENDER SYSTEMS*, 2001.
- [Stefanidis et al.2009] Stefanidis K., Drosou M., Pitoura E., « "You May Also Like" Results in Relational Databases », *Proc. of the 3rd International Workshop on Personalized Access, Profile Management and Context Awareness : Databases (PersDB 2009), in conjunction with the VLDB 2009 Conference*, 2009.
- [White2007] White R. W., « Studying the use of popular destinations to enhance Web search interaction », *ACM SIGIR '07. ACM*, Press, 2007, p. 159–166.
- [Yang et al.2009] Yang X., Procopiuc C. M., Srivastava D., « Recommending Join Queries via Query Log Analysis », *ICDE*, 2009, p. 964-975.